Europäisches Patentamt
European Patent Office
Office européen des brevets

AT04/408

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

03450257.5

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 03450257.5
Demande no:

Anmeldetag:
Date of filing: 21.11.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Neswal, Peter
Lassallestrasse 14
2231 Strasshof an der Nordbahn
AUTRICHE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Verfahren zur Installation und Konfiguration von Softwarekomponenten

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F9/40

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PT RO SE SI SK TR LI

Verfahren zur Installation und Konfiguration von Softwarekomponenten

Die vorliegende Erfindung betrifft ein Verfahren zur automatischen Installation und Konfiguration von Softwarekomponenten in einem Computernetzwerk, das eine Vielzahl von Benutzerrechnern und zumindest eine Netzwerkressource von installierbaren Softwarekomponenten umfaßt, wobei die erfolgreiche Installation einer Softwarekomponente die An- oder Abwesenheit einer anderen Softwarekomponente voraussetzen kann. Die Erfindung betrifft ferner Computerprogrammobjekte zur Durchführung dieses Verfahrens in Form eines Regelpakets, eines Frameworks und eines Klientprogramms, sowie Computer und Datenträger, die mit solchen Programmobjekten programmiert sind.

Die Verteilung, Installation und Konfiguration von Softwarekomponenten in größeren Computernetzwerken, z.B. Firmennetzwerken mit Tausenden von unterschiedlichen Benutzerrechnern, auf denen Hunderte von Softwarekomponenten laufen, die überdies teils unterschiedlich zu konfigurieren sind, ist in der Praxis ein nicht-triviales Problem.

Zur Lösung dieses Problems wurden bereits verschiedenste Mechanismen vorgeschlagen, denen allen gemeinsam ist, daß sie von einer zentralen Verteilung der Softwarekomponenten ausgehen. Dies erfordert einerseits genaue Kenntnis über die Ausstattung und Anforderungsprofile aller Benutzerrechner im Feld, was den Aufbau und die Verwaltung umfangreicher Verzeichnisse und Verteilungsschlüssel bedeutet. Andererseits können zentralen Verteilungsstrategien nicht auf rasche Vor-Ort-Veränderungen der Hard- oder Softwareausstattung oder der Konfiguration eines Benutzerrechners reagieren, wie das Anschließen neuer Hardware, das Anmelden in einem Netzwerk, das Anmelden eines Benutzers usw.

Die Erfindung setzt sich zum Ziel, Verfahren, Programmobjekte und gemäß diesen programmierte Computer und Datenträger zu schaffen, mit denen Softwarekomponenten in einem großmaßstä-

bigen Computernetzwerk verteilt, installiert und individuell konfiguriert werden können, und dies mit minimalem Verwaltungsaufwand, größtmöglicher Flexibilität und rascher Reaktion auf lokale Änderungsereignisse an den Benutzerrechnern.

Dieses Ziel wird in einem ersten Aspekt mit einem Verfahren der einleitend genannten Art erreicht, das sich gemäß der Erfindung auszeichnet durch die Schritte:

a) Bereitstellen eines Frameworks auf der Netzwerkressource, welches ein Regelpaket für jede installierbare Softwarekomponente, einen Detektor für jede mögliche Voraussetzung und eine Liste abzuarbeitender Regelpakete umfaßt,

wobei jedes Regelpaket zumindest eine der folgenden vier Routinen umfaßt: eine Routine zum Installieren der Softwarekomponente auf dem Benutzerrechner, eine Routine zum Deinstallieren derselben, eine Routine zum Konfigurieren der installierten Softwarekomponente und eine Routine zum Rückgängigmachen (Dekonfigurieren) der Konfiguration hinsichtlich derselben;

b) Übertragen des Frameworks an einen Benutzerrechner;

c) Abarbeiten der Liste abzuarbeitender Regelpakete auf dem Benutzerrechner unter Aufrufen ihrer Installationsroutinen, und nochmaliges Abarbeiten der Liste abzuarbeitender Regelpakete auf dem Benutzerrechner unter Aufrufen ihrer Konfigurationsroutinen;

wobei, wenn im Zuge eines Regelpakets mittels eines Detektors festgestellt wird, daß die An- oder Abwesenheit einer anderen Softwarekomponente erforderlich ist, die Installations- bzw. Deinstallationsroutine des dieser anderen Softwarekomponente zugeordneten Regelpakets aufgerufen wird.

Auf diese Weise wird erstmals eine automatisierte, dezentrale und dynamische Selbstinstallation und -konfiguration jedes einzelnen Benutzerrechners möglich, welche rasch auf lokale Ereignisse reagieren kann. Jeder Benutzerrechner erhält ein Framework aus Regeln in Form von Regelpaketen, die allen potentiell installierbaren Softwarepaketen zugeordnet sind. Dadurch wird eine zentrale Verteilung von Softwarekomponenten mit all

den beschriebenen Nachteilen vermieden; lediglich eine Abbildung der gegenseitigen Abhängigkeiten aller installierbaren Softwarekomponenten wird in Form des Frameworks verteilt.

Die Sammlung und Wartung zentraler Verzeichnisse über die Ausstattung und Konfiguration aller Benutzerrechner entfällt. Der Aufbau des erfindungsgemäßen Frameworks ist wesentlich einfacher, da hier nur einmal für jede installierbare Softwarekomponente ihr Einsatzgebiet und ihre Anforderungen zu definieren sind. Dazu stellt das Framework auch wiederverwendbare Detektoren zur Verfügung, mit deren Hilfe die Voraussetzungen für die Installation oder Konfiguration einer Softwarekomponente auf dem Benutzerrechner rasch überprüft werden können.

Die Regelpakete für die einzelnen Softwarekomponenten sind ausschließlich über ihre gegenseitigen Abhängigkeiten referenziert, sonst aber autark. Dies erleichtert einerseits die Definition der Regelpakete für die Bereitstellung des Frameworks, andererseits brauchen die Regelpakete auf dem Benutzerrechner nur eines nach dem anderen abgearbeitet zu werden, wobei sie sich entsprechend ihrer Abhängigkeiten auch gegenseitig aufrufen können. Jedes Regelpaket „weiß" selbst, wie es seine zugeordnete Softwarekomponente installieren, deinstallieren, konfigurieren bzw. dekonfigurieren kann. Das Erzeugen von speziellen Installations- oder Konfigurations-Scripts für die einzelnen Benutzerrechner entfällt.

Mit dem Verfahren der Erfindung ist nicht nur die Verteilung und die Installation von Softwarekomponenten möglich, sondern gleichzeitig auch ihre Konfiguration, d.h. die Einstellung spezieller Parameter der installierten Softwarekomponente. Damit können z.B. benutzerspezifische, anwendungsumgebungsspezifische, gruppenspezifische oder aber auch einfach nur firmeneinheitliche Konfigurationen auf allen Benutzerrechnern im Netzwerk durchgeführt werden. Alles, was dazu erforderlich ist, ist die einmalige Definition eines Regelpakets für die jeweilige Softwarekomponente.

Eine bevorzugte Ausführungsform des Verfahrens der Erfindung zeichnet sich dadurch aus, daß das Framework auch Detektoren für die Hardware- oder Betriebssystemausstattung eines Benutzerrechners umfaßt und im Zuge einer Routine mittels eines solchen Detektors überprüft wird, ob der Benutzerrechner für die jeweilige Installation, Deinstallation, Konfiguration oder Dekonfiguration der Softwarekomponente geeignet ist, und/oder daß im Zuge einer Routine vorab geprüft wird, ob die jeweilige Installation, Deinstallation, Konfiguration oder Dekonfiguration der Softwarekomponente auf dem Benutzerrechner bereits erfolgt ist und, wenn ja, die Routine sofort beendet wird.

Dadurch wird jedes Regelpaket noch stärker autark, d.h. es „weiß" auch, ob es für den jeweiligen Benutzerrechner zuständig bzw. anzuwenden ist. Die Abarbeitung der Regelpakete auf den Benutzerrechnern wird dadurch noch einfacher. Im allgemeinsten Fall können z.B. einfach alle im Framework vorhandenen Regelpakete abgearbeitet werden, beginnend mit dem ersten, und jedes Regelpaket entscheidet für sich, ob es überhaupt auszuführen ist, andere, vorauszusetzende Regelpakete aufrufen soll usw.

Wie bereits erörtert, besteht ein entscheidender Vorteil des dezentralen Verfahrens der Erfindung darin, daß es besonders rasch auf lokale Änderungsereignisse am Benutzerrechner reagieren kann. Zu diesem Zweck wird bevorzugt vorgesehen, daß Schritt b) und/oder Schritt c) durch ein lokales Ereignis auf dem jeweiligen Benutzerrechner ausgelöst wird, z.B. ein Systemereignis wie ein Systemstart oder -stop, eine Benutzeran- oder -abmeldung, eine Netzwerkan- oder -abmeldung, ein Programmstart oder -ende usw., ein Hardwareereignis wie das An- oder Abschließen einer Hardwareausstattung, usw.

Grundsätzlich ist es auch möglich, daß alternativ oder zusätzlich Schritt b) und/oder Schritt c) durch ein entferntes Ereignis auf der Netzwerkressource ausgelöst wird, z.B. das Aussenden einer Gruppen- oder Broadcastnachricht usw., wodurch auch das Verhalten herkömmlicher zentraler Verteilungsverfahren mit dem erfindungsgemäßen Verfahren nachgebildet werden kann.

Die Erfindung erstreckt sich auch auf ein Computerprc gramm, welches das erfindungsgemäße Verfahren implementiert.

Ein weiterer Aspekt der Erfindung besteht in der Schaffun eines Regelpakets, das auf einem Betriebssystem eines Benutzer rechners ausführbar ist, wie in den Ansprüchen 7 bis 11 defi niert. Bezüglich der Merkmale und Vorteile des erfindungsgemä ßen Regelpakets wird auf die obigen Ausführungen zum Verfahre verwiesen.

Eine bevorzugte Ausführungsform eines Regelpakets der Er findung zeichnet sich dadurch aus, daß es mindestens einen Aus löseverweis auf ein lokales Ereignis auf dem Benutzerrechne oder ein entferntes Ereignis auf der Netzwerkressource enthält wobei der Auslöseverweis diesem Ereignis zumindest eine de Routinen des Regelpakets zuordnet. Dadurch können auch einzeln Regelpakete bzw. deren Routinen ereignisgesteuert ausgeführ werden, was die Flexibilität und Reaktionsfähigkeit des System wesentlich erhöht.

In der Praxis kommen ständig neue Softwarekomponenten au den Markt. Wenn keine besonderen Maßnahmen ergriffen werden würde die Anzahl der Regelpakete im Framework ständig anwach sen; andererseits werden Regelpakete im Framework obsolet, z.B wenn Softwarekomponenten außer Dienst gestellt werden. Solch obsoleten Regelpakete werden zweckmäßigerweise aus dem Frame work entfernt, auf einzelnen Benutzerrechnern können sie jedoc noch erforderlich sein, beispielsweise wegen veralteter Hard warekomponenten. Es ist daher besonders günstig, wenn Regelpa kete auch in einen inaktiven Zustand versetzbar sind, in wel chem nur ihre Deinstallationsroutine aufrufbar ist. Dadurc kann die Installation veralteter Softwarekomponenten verhinder werden, ihre Deinstallation ist aber jederzeit möglich.

Die Erfindung erstreckt sich auch auf einen Computer, de mit zumindest einem erfindungsgemäßen Regelpaket programmier ist.

Ein weiterer Aspekt der Erfindung ist ein Framework, da auf einer Netzwerkressource in einem Computernetzwerk für ein

Vielzahl von Benutzerrechnern bereitstellbar ist, wie in den Ansprüchen 13 und 14 definiert, und das erfindungsgemäße Regelpakete enthält. Bezüglich der Merkmale und Vorteile des Frameworks wird auf die obigen Ausführungen zum Verfahren verwiesen.

Die Erfindung erstreckt sich auch auf einen Computer und einen maschinenlesbaren Datenträger, die mit einem erfindungsgemäßen Framework programmiert sind.

Noch ein weiterer Aspekt der Erfindung besteht in der Schaffung eines Klientprogramms, das auf einem Benutzerrechner ausführbar ist, wie in den Ansprüchen 17 bis 21 definiert, und ein Framework gemäß der Erfindung enthält. Bezüglich weiterer Merkmale und Vorteile des Klientprogramms wird auf die obigen Ausführungen zum Verfahren verwiesen.

Gemäß einer bevorzugten Ausführungsform weist das Klientprogramm eine lokale Datenbank auf, welche eine Liste von Regelpaketen mit erfolgreich durchlaufenen Installationsroutinen und eine Liste von Regelpaketen mit erfolgreich durchlaufenen Konfigurationsroutinen enthält. Mit Hilfe dieser Datenbank kann die Abarbeitung der Regelpakete beschleunigt werden, da beispielsweise für bereits installierte oder konfigurierte Softwarepakete der Aufruf der entsprechenden Regelpakete unterbleiben kann.

Überdies eröffnet dies die Möglichkeit, daß das Klientprogramm die in den Listen eingetragenen Regelpakete mit den im Framework enthaltenen Regelpaketen vergleicht und für jene Regelpakete, welche im Framework nicht aufscheinen, in einem ersten Durchgang deren Dekonfigurationsroutinen und in einem zweiten Durchgang deren Deinstallationsroutinen abarbeitet, wodurch obsolete oder veraltete Softwarekomponenten automatisch entfernt werden können.

Gemäß einer bevorzugten Ausführungsform eines Klientprogramms, welches von Regelpaketen mit Auslöseverweisen zur ereignisgesteuerten Ausführung Gebrauch macht, überwacht das Klientprogramm das Auftreten eines lokalen Ereignisses auf dem Benutzerrechner, z.B. eines Systemereignisses wie eines System-

starts oder -stops, einer Benutzeran- oder -abmeldung, eine Netzwerkan- oder -abmeldung, eines Programmstarts oder -ende usw., eines Hardwareereignisses wie des An- oder Abschließen einer Hardwareausstattung, usw., und/oder eines entfernten Ereignisses auf der Netzwerkressource, z.B. des Aussendens einer Gruppen- oder Broadcastnachricht usw., und ruft die diesem Ereignis über den Auslöseverweis zugeordnete Routine des entsprechenden Regelpakets auf. Dies kann zweckmäßigerweise auch mit Hilfe der Listen in der Datenbank erfolgen, in welche die Auslöseverweise der Regelpakete miteingetragen werden können. Auf diese Weise können auch einzelne oder Gruppen von Regelpaketen bzw. deren Routinen ereignisgesteuert ausgeführt werden.

In jedem Fall ist es besonders günstig, wenn das Klientprogramm ein Transaktionssystem für jede systemmodifizierende Komponente, insbesondere für die Regelpakete, aufweist. Dadurch kann jederzeit ein Rollback des Systems vorgenommen werden, wenn z.B. eine Installation oder Konfiguration fehlschlägt, wie in der Technik bekannt. Die Betriebssicherheit des Klientprogramms wird dadurch wesentlich erhöht.

Schließlich erstreckt sich die Erfindung auch auf einen Computer, der mit einem erfindungsgemäßen Klientprogramm programmiert ist.

Die Erfindung wird nachstehend anhand von in den Zeichnungen dargestellten Ausführungsbeispielen näher erläutert. In der Zeichnungen zeigt:

Fig. 1 ein Blockschaltbild eines Computernetzwerkes, in welchem das Verfahren, die Programmobjekte und Computer der Erfindung zur Anwendung gelangen,

Fig. 2 ein Blockschaltbild eines beispielhaften, mit einem Klientprogramm programmierten Benutzerrechners der Erfindung,

Fig. 3 den schematischen Aufbau eines Frameworks der Erfindung,

Fig. 4 den schematischen Aufbau eines Regelpakets der Erfindung,

Fig. 5 die gegenseitigen Beziehungen mehrerer beispielhafter Regelpakete in Form eines Beziehungsdiagramms,

Fig. 6 ein Flußdiagramm des Verfahrens der Erfindung,

Fig. 7 ein Beispiel für die von einer Installationsroutine erzeugten Einträge in der lokalen Datenbank,

Fig. 8 ein Beispiel für die von einer Konfigurationsroutine erzeugten Einträge in der lokalen Datenbank,

Fig. 9 mehrere mögliche Auslösearten für die auf dem Benutzerrechner ablaufenden Schritte des Verfahrens der Erfindung in Form eines Flußdiagramms, und

Fig. 10 das Blockschaltbild einer möglichen Implementierung des Klientprogramms auf einem Betriebssystem mit voneinander isolierten Ausführungsschichten.

Fig. 1 zeigt ein Computernetzwerk 1, das eine Vielzahl von Benutzerrechnern 2 umfaßt. Ein beispielhafter Benutzerrechner 2 ist in Fig. 2 ausführlicher gezeigt und enthält eine Anzahl von schematisch dargestellten Softwarekomponenten BS1, A, B,.., welche je nach Einsatzgebiet des Benutzerrechners 2 rechner-, benutzer- oder anwendungsspezifisch installiert und konfiguriert werden sollen.

Die Gesamtheit aller auf dem Benutzerrechner 2 potentiell installier- und konfigurierbarer Softwarekomponenten ist in Fig. 2 mit SW bezeichnet. Dabei ist zu beachten, daß die Installation und Konfiguration der Softwarekomponenten SW komplexen gegenseitigen Abhängigkeiten unterworfen sein kann. Beispielsweise erfordert die Installation der Softwarekomponente B eine vorherige Installation der Softwarekomponente A und diese wiederum eine vorherige Installation der Softwarekomponente BS1, welche z.B. bereits auf dem Benutzerrechner 2 installiert sein kann, weil sie Teil des Betriebssystems ist. Andererseits kann es Softwarekomponenten geben, welche unbedingt die Abwesenheit, d.h. Deinstallation, einer anderen Softwarekomponenten für ihre korrekte Installation voraussetzen. Eine solche Situation ist in Fig. 5 dargestellt (welche später noch ausführlicher erörtert wird), wobei die mit „restrict" bezeichneten

Pfade zwischen Softwarekomponenten die Voraussetzung der Abwe senheit einer Softwarekomponente angeben, jene mit „require bezeichneten Pfade die Voraussetzung der Anwesenheit eine Softwarekomponente.

Es versteht sich, daß der Begriff „Softwarekomponente" wie er hier verwendet wird, je nach Anwendungsfall und Erfor dernis jede Art von Granularität bzw. „Korngröße" von Softwar umfaßt, sei es ein Treiber, ein Unterprogramm, ein Programmob jekt, ein Hauptprogramm, eine Unter- oder Hauptklasse, eine An wendung, oder ein Anwendungskomplex. Darin zeigt sich die Mäch tigkeit der hier vorgestellten Lösung: So kann z.B. ein Regel paket RP_1 für die allgemein bekannte Softwarekomponente „Micrc soft Office XP mit Microsoft Frontpage, ohne Netzwerkunterstüt zung" definiert werden, gleichzeitig ein weiteres Regelpake RP_2 für die - sich teilweise überlappende, teilweise unterfal lende - Softwarekomponente „Microsoft Frontpage, mit Netzwerk unterstützung".

Zurückkehrend auf Fig. 1 wird das gesamte Beziehungssyste aller potentiell auf den Benutzerrechnern 2 installierbare Softwarekomponenten SW in Form eines Frameworks FW abgebildet dessen struktureller Aufbau später noch ausführlicher unter Be zugnahme auf die Fig. 3 und 4 erläutert wird. Das Framework F wird im Computernetzwerk 1 auf einer Netzwerkressource RES_1 be reitgestellt.

Unabhängig von dem Framework FW werden im Computernetzwer 1 auf einer weiteren Netzwerkressource RES_2 alle potentiell in stallierbaren Softwarekomponenten SW (BS1, A, B,..) bereitge stellt.

Es versteht sich, daß die Netzwerkressourcen RES_1 und RE auch ein und dieselbe Netzwerkressource RES sein können (sieh z.B. Fig. 2) oder ihrerseits in geographisch verteilte Netz werkressourcen aufgeteilt werden können (siehe z.B. die Vertei lung der Softwarekomponenten A und B auf die Netzwerkressource RES_2 und RES_3 in Fig. 2). Auch ist es möglich, daß eine de Netzwerkressourcen, insbesondere jene für die Softwarekomponen

ten, tatsächlich ein „offline"-Datenträger ist, z.B. eine CD-Rom usw., wie in Fig. 2 beispielhaft für die Softwarekomponente B angedeutet.

Der Aufbau und die Pflege des Frameworks FW, d.h. das Einspeisen in die Netzwerkressource RES_1, werden an Administratorarbeitsplätzen 3 durchgeführt. Die Verteilung bzw. Ausbreitung des Frameworks FW im Computernetzwerk 1 bis zu den Benutzerrechnern 2 kann auf jede beliebige Art erfolgen, beispielsweise durch Push-Technologien, wie Broadcast- oder Gruppennachrichten nach dem Internetprotokoll, oder Pull-Technologien, wie Abholen durch die Benutzerrechner 2 von Logon-Shares auf den Netzwerkressourcen bei einem Systemstart oder einer Benutzeranmeldung, durch Peer-to-Peer-Propagierungs- oder Replizierungsverfahren usw.

Beispielsweise kann das Framework FW in der Art von Internet-Domain-Name-Services von Netzwerkknoten, wie der Netzwerkressource RES_1, zu Netzwerkknoten, wie der Netzwerkressource RES_2, mittels Spiegelung repliziert und damit verteilt werden. Dadurch kann das Framework FW auch auf Netzwerkressourcen RES_2 verfügbar sein, welche für die Bereithaltung von Softwarekomponenten SW dienen, oder auch direkt von Benutzerrechner 2 zu Benutzerrechner 2 („Peer-to-Peer") repliziert werden. Um den Netzwerkverkehr bei der Verteilung des Frameworks FW möglichst gering zu halten, kann auch vorgesehen werden, daß nach einer erstmaligen Verteilung des gesamten Frameworks FW nur mehr Differenz-Updates des Frameworks FW auf seine jeweils aktuellste Version verteilt werden.

Die weitere Beschreibung der Erfindung geht von einem Zustand aus, in welchem das Framework FW dem stellvertretend beschriebenen Benutzerrechner 2 zur Verfügung steht.

Der Aufbau des Frameworks FW wird anhand der Fig. 3 und 4 näher erläutert. Gemäß Fig. 3 umfaßt das Framework FW einen Satz von Regelpaketen RP, und zwar ein Regelpaket RP für jede potentiell auf einem Benutzerrechner 2 installier- und/oder konfigurierbare Softwarekomponente BS1, A, B,.. Jedes Regelpa-

ket RP bildet die Hard- und Softwarevoraussetzungen jeweils ei ner Softwarekomponente ab.

Fig. 4 zeigt den Aufbau eines beispielhaften Regelpaket RP_A für die Softwarekomponente A. Das Regelpaket RP_A enthäl einen Verweis RES_A auf seine zugeordnete Softwarekomponente A beispielsweise in Form eines Zeigers auf jene Netzwerkressourc RES_2, auf welcher die Softwarekomponente A verfügbar ist.

Gemäß Fig. 4 umfaßt jedes Regelpaket RP eine Routin „INST()" 4 zum Installieren der ihm zugeordneten Softwarekompo nente (hier: A) auf dem Benutzerrechner 2, eine weitere Routin „DEINST()" 4′ zum Deinstallieren diese Softwarekomponente vo Benutzerrechner 2, eine Routine „CONFIG()" 5 zum Konfiguriere dieser Softwarekomponente und eine Routine „DECONFIG()" 5′ zu Rückgängigmachen („Dekonfigurieren") der Konfiguration diese Softwarekomponente.

Ein Regelpaket RP muß nicht alle vier Routinen 4, 4′, 5 5′ enthalten, jedoch mindestens eine der Routinen 4, 4′, 5, 5′ Bevorzugt enthält das Regelpaket RP mindestens ein komplementä res Routinenpaar 4/4′ oder 5/5′, so daß zu der Installations bzw. Konfigurationsroutine 4, 5 jeweils zumindest die zugeord nete Deinstallations- bzw. Dekonfigurationsroutine 4′, 5′ ent halten ist.

Es versteht sich, daß die vier Routinen 4, 4′, 5 und 5′ soferne sie gemeinsame Codeabschnitte besitzen - auch strecken weise in einer gemeinsamen Routine zusammengefaßt sein können z.B. in einer gemeinsam zu durchlaufenden Einleitungsroutin des Regelpakets RP, oder überhaupt durch einen gemeinsamen Co deabschnitt implementiert werden können, welcher durch entspre chende Aufrufschalter gesteuert einmal z.B. die Funktion de Installationsroutine 4, ein anderes Mal z.B. die Funktion de Dekonfigurationsroutine 5′ einnimmt, usw. In diesem Sinne sin die Routinen 4, 4′, 5, 5′ „funktionelle" Routinen, nicht not wendigerweise Routinen im programmtechnischen Sinne, wie fü den Fachmann ersichtlich.

In der vorliegenden Beschreibung wird der Begriff „Installieren" zum grundsätzlichen Bereitstellen der Nutzungsmöglichkeit einer Softwarekomponente auf einem Benutzerrechner verwendet. Die Installation umfaßt in der Regel das Speichern der Softwarekomponente A auf einem lokalen Datenspeicher (z.B. der Festplatte) des Benutzerrechners, sowie häufig auch ein erstes, allgemeines Konfigurieren (siehe unten) der Softwarekomponente, damit diese grundsätzlich betriebsfähig ist.

Der Begriff „Deinstallieren" wird für das Entfernen der Softwarekomponente SW vom Benutzerrechner 2, z.B. durch Löschen von der Festplatte, verwendet.

Der Begriff „Konfigurieren" wird wiederum für jede Form von einheitlicher, gruppenspezifischer, benutzerspezifischer oder anwendungssituationsspezifischer Einstellung einer Softwarekomponente verwendet, wie durch den Einstellungspfeil in Fig. 2 versinnbildlicht. Damit ermöglicht das erfindungsgemäße Verfahren nicht nur die autarke Installation aller erforderlichen Softwarekomponenten auf einem Benutzerrechner, sondern auch die Einstellung eines bestimmten, im Framework FW definierten Zustandes dieser Softwarekomponenten.

Unter dem Begriff „Rückgängigmachen einer Konfiguration" bzw. „Dekonfigurieren" wird in der vorliegenden Beschreibung das Wiederherstellen jener Einstellung einer Softwarekomponente verstanden, wie sie vor dem Konfigurieren geherrscht hat, und/oder das Einstellen der nach einer Deinstallation dieser Softwarekomponente verbliebenen anderen Softwarekomponenten in der Art, wie es für den aktuellen Systemzustand ohne die deinstallierte Softwarekomponente erforderlich ist; letzteres ist auch mit dem Begriff Rückgängigmachen der Konfiguration „hinsichtlich" dieser Softwarekomponente gemeint.

Gemäß Fig. 4 kann das Regelpaket RP_A optional einen oder mehrere Auslöseverweise $TRIG_A$ auf ein lokales oder entferntes Ereignis enthalten, bei dessen Auftreten zumindest eine seiner Routinen 4, 4', 5, 5' ausgeführt werden soll, wie später noch anhand von Fig. 9 ausführlicher erörtert wird.

Ferner kann das Regelpaket RP_A auch lokale Ressource RES_4, RES_5 für z.B. kleine Softwarekomponenten oder Konfigurationsparameter umfassen.

Jede der Routinen 4, 4', 5, 5' enthält eine eigenständig Überprüfung der Voraussetzungen für die Installation, Konfiguration, Deinstallation oder Dekonfiguration der dem Regelpake zugeordneten Softwarekomponente, beispielsweise das Erforderni der Anwesenheit einer anderen Softwarekomponente („require" Pfade in Fig. 5) oder der Abwesenheit einer Komponente („restrict"-Pfade in Fig. 5). Wenn die jeweilige Routine ein Anwesenheitserfordernis („require") feststellt, ruft sie die Installationsroutine 4 des dieser anderen Softwarekomponente zugeordneten anderen Regelpakets RP auf; wenn sie ein Abwesenheitserfordernis („restrict") feststellt, dessen Deinstallationsroutine 4'. Auf diese Weise wird durch Abarbeiten der Regelpakete das Beziehungsgeflecht von Fig. 5 eingehalten und ausgeführt. Es ist klar, daß diese Überprüfung auch in einen de Routinen gemeinsamen Teil des Regelpakets ausgelagert sei kann, welcher bei Ausführung einer Routine stets durchlaufe wird.

Um die Überprüfung der An- oder Abwesenheit einer bestimmten Softwarekomponente zu vereinfachen, umfaßt das Framework F einen Satz von Detektionsroutinen bzw. Detektoren DET, z.B. einen Detektor DET_A für das Vorhandensein der Softwarekomponent A, einen Detektor DET_{BS1} für das Vorhandensein der Betriebssystemkomponente BS1 oder einen Detektor DET_{HW} für das Vorhandensein einer bestimmten Hardwareausstattung des bestimmten Benutzerrechners 2, siehe Fig. 5.

Die Detektoren DET können nicht nur die An- oder Abwesenheit einer Softwarekomponente SW überprüfen, sondern auch ob eine Softwarekomponente gerade läuft bzw. ausgeführt wird oder nicht. Alle diese Varianten werden in der vorliegenden Beschreibung unter dem Begriff „An- oder Abwesenheit" zusammengefaßt bzw. sind eine der möglichen Voraussetzungen für eine Softwarekomponente, die ein Detektor DET überprüfen kann.

Die Detektoren DET können sich auch gegenseitig aufrufen bzw. voneinander Gebrauch machen, z.B. wenn eine zu überprüfende Voraussetzung in mehrere einzelne Voraussetzungen zerlegbar ist, für welche bereits andere Detektoren vorhanden sind.

Anlage 1 zeigt ein Implementierungsbeispiel für einen Detektor zur Feststellung der Anwesenheit der Softwarekomponente „Microsoft Word 10.0" der Firma Microsoft. Die Subroutine „query_exist" überprüft die Anwesenheit der Softwarekomponente; die Subroutine „query_active", ob die Softwarekomponente läuft.

Jede der Routinen 4, 4', 5, 5' kann vorteilhafterweise so gestaltet werden, daß sie selbst bzw. mittels entsprechender Detektoren DET überprüft, ob sie für die auf dem Benutzerrechner 2 vorgefundene Hardware- oder Softwareausstattung geeignet ist und, wenn nicht, beispielsweise ohne weitere Aktion beendet wird, d.h. die Steuerung zurückgibt. Auch kann die Routine überprüfen, ob die aufgerufene Installation, Deinstallation, Konfiguration oder Dekonfiguration ihrer Softwarekomponente nicht ohnehin bereits erfolgt ist, in welchem Fall sie ebenfalls beendet wird, d.h. die Steuerung zurückgibt. Alternativ könnten diese Prüfungen aber auch in einen den Routinen gemeinsamen Codeabschnitt (siehe oben) des Regelpakets RP oder in der aufrufenden Prozeß P (siehe unten) verlagert sein.

Schließlich umfaßt das Framework FW eine Liste L jener Regelpakete RP, mit deren Abarbeitung im Benutzerrechner 2 begonnen werden soll. Es versteht sich, daß die Liste L z.B. nur auf das erste Regelpaket RP verweisen kann, welches in weitere Regelpakete RP rekursiert, oder einfach alle Regelpakete RP anführt, wenn diese jeweils für sich die Voraussetzungen für ihre Ausführung feststellen, oder aber nur solche Regelpakete anführt, die von einem Administrator als „Tagespensum" vorgegeben werden usw.

In einer bevorzugten Implementierung besteht ein Regelpaket RP aus einem Satz von Dateien, die durch eine zentrale Regelpaket-Spezifizierungsdatei referenziert werden. Ein Beispiel einer solchen Regelpaket-Spezifizierungsdatei ist in Anlage 2

angeführt. Im Einleitungsabschnitt der Datei ist der Verwei auf die Softwarekomponente ersichtlich; die Verweise „install" „uninstall", „policies_true" und „policies_false" zeigen au die vier Routinen 4, 4', 5 bzw. 5'.

Die Auswertung des Frameworks FW und Abarbeitung der Re gelpakete RP auf dem Benutzerrechner 2 wird mit Hilfe eine Klientprogramms KP durchgeführt, welches die beschriebene Abar beitung der Liste L in einem Prozeß P durchführt (Fig. 2). Z diesem Zweck enthält das Klientprogramm KP einen Speicher S zu lokalen Speicherung einer Kopie des Frameworks FW, welcher auc für die Weiterverteilung (Replikation) des Frameworks FW an an dere Benutzerrechner 2 verwendet werden kann.

Das Klientprogramm KP verfügt ferner über eine lokale Da tenbank DB, welche zwei Listen 7, 8 führt, deren Verwendung i den Fig. 7 und 8 ausführlicher gezeigt ist. Die erste Liste enthält einen Eintrag für jedes Regelpaket RP, dessen Installa tionsroutine 4 erfolgreich durchlaufen wurde. Die zweite List 8 enthält einen Eintrag für jedes Regelpaket RP, dessen Konfi gurationsroutine 5 erfolgreich durchlaufen wurde. Damit verfüg das Klientprogramm KP über eine Aufstellung aller auf dem Be nutzerrechner 2 erfolgreich installierten und konfiguriertei Softwarekomponenten SW, welche bei der Abarbeitung der Regelpa kete RP auch dazu verwendet werden kann, Doppelaufrufe ode endlose Rekursionen usw. zu erkennen und zu vermeiden. Darübe hinaus ist die lokale Datenbank DB auch nützlich, um obsolet oder veraltete Softwarekomponenten zu erkennen und zu entfer nen, wie im Rahmen des Flußdiagramms von Fig. 6 noch ausführ lich erläutert wird.

Fig. 6 zeigt ein beispielhaftes Flußdiagramm für da Klientprogramm KP. Nach einer Initialisierung im Block 9 werde im Block 10 alle in der Liste L des Frameworks FW angeführte Regelpakete RP abgearbeitet, und zwar durch Aufrufen ihrer In stallationsroutinen 4. Es versteht sich, daß dabei die Regelpa kete RP, wenn sie mittels der Detektoren DET die Voraussetzun der Anwesenheit oder Abwesenheit anderer Regelpakete RP fest

stellen, jeweils in diese anderen Regelpakete rekursieren, wie bereits erläutert.

Nachdem im Block 10 alle Installationsroutinen 4 abgearbeitet und damit alle erforderlichen Softwarekomponenten BS1, A, B,.. auf dem Benutzerrechner 2 installiert worden sind, geht das Klientprogramm KP bzw. sein Prozeß P zum Block 11 über, in welchem die Konfiguration der Softwarepakete in einem zweiten Durchlauf durch die Liste L erfolgt. Im Block 11 werden wieder die in der Liste L angeführten Regelpakete RP abgearbeitet, diesmal unter Aufrufen ihrer Konfigurationsroutine 5. Falls erforderlich, können die Regelpakete RP wieder in weitere Regelpakete rekursieren, wie bereits erörtert.

Dadurch, daß im Block 10 zunächst eine vollständige Installation aller erforderlichen Softwarekomponenten erfolgt, wird gewährleistet, daß das Konfigurieren im Block 11 von einem definierten Systemzustand ausgeht, was in vielen Fällen eine korrekte Konfiguration erst ermöglicht.

Die weiteren in Fig. 6 gezeigten Blöcke 12 und 13 des Verfahrens bzw. des Klientprogramms KP sind optional und dienen der Beseitigung obsoleter oder veralteter Softwarekomponenten vom Benutzerrechner 2.

Wie bereits erwähnt, sind Regelpakete RP für obsolete oder veraltete Softwarekomponenten nicht mehr im Framework FW enthalten, können allerdings auf einem Benutzerrechner 2 noch erforderlich sein, beispielsweise wegen veralteter Hardwareausstattung. Das Klientprogramm KP vergleicht daher die im Framework FW enthaltenen Regelpakete RP mit den in den Listen 7 und 8 der lokalen Datenbank DB eingetragenen Regelpaketen RP und versetzt jene Regelpakete RP, welche im Framework FW nicht mehr aufscheinen, in einen inaktiven Zustand, z.B. durch ein entsprechendes Flag in den Listen 7 und 8 oder im Regelpaket RP selbst. Im inaktiven Zustand ist ein Regelpaket RP nur mehr anhand seiner Deinstallationsroutine 4' oder seiner Dekonfigurationsroutine 5' aufrufbar.

Im Block 12 werden alle inaktiven Regelpakete RP anhan ihrer Dekonfigurationsroutinen 5' aufgerufen. Im anschließende Block 13 erfolgt ein erneuter Durchgang anhand ihrer Deinstal lationsroutinen 4'.

Wie bereits erörtert, prüft dabei jede Deinstallations oder Dekonfigurationsroutine (oder auch der Prozeß P), ob si auf ihr „Ziel", den Benutzerrechner 2, anwendbar ist und nu dann, wenn dies möglich ist (z.B. Ausbau einer veralteten Hard ware), wird sie ausgeführt. Bei der Dekonfiguration bzw. Dein stallation trägt sie sich auch jeweils wieder aus der Liste bzw. 8 der lokalen Datenbank DB aus.

Dadurch wird bei jedem Durchlauf des Klientprogramms K gleichsam ein Reinigungsdurchlauf ausgeführt, der veraltet oder obsolete Softwarekomponenten und deren Regelpakete besei tigt.

Im Block 14 des Flußdiagramms von Fig. 6 werden Abschluß prozesse durchgeführt und das Klientprogramm KP beendet.

Das Ausführen des Klientprogramms KP auf dem Benutzerrech ner 2 kann auf vielerlei Arten angestoßen werden. Fig. 9 zeig einige mögliche Varianten. Dem Abarbeitungsprozeß P des Klient programms KP ist hier ein Ereignismanager 15 vorgeschaltet welcher sowohl lokale Ereignisse auf dem Benutzerrechner 2 al: auch entfernte Ereignisse z.B. auf den Wartungsarbeitsplätzen verarbeiten kann.

Sinnbildlich sind einige Arten von lokalen Ereignissen dargestellt, beispielsweise Benutzerereignisse 16 wie die Betä tigung einer entsprechenden Taste, Systemereignisse 17 wie da: Erkennen eines Systemstarts oder -stops, einer Benutzeran- ode: -abmeldung, einer Netzwerkan- oder -abmeldung, eines Programm starts oder -endes usw., Hardwareereignisse 18 wie das An- ode: Abschließen einer Hardwareausstattung, oder vom Systemadmini strator definierte lokale Ereignisse 19. Es ist aber auch mög lich, daß das Klientprogramm KP durch entfernte Ereignisse aus gelöst wird, beispielsweise durch aktives Senden eines Trigger befehls von einer Wartungsarbeitsstation 3 her, oder durcl

„passives" Abholen eines Auslösebefehls von einer Netzwerkressource RES_1, z.B. im Zuge einer Netzwerkanmeldung oder zu vorgegebenen Tageszeiten.

Die genannten Ereignisse können auch direkt die Ausführung bestimmter Regelpakete RP oder Routinen 4, 4', 5, 5' auslösen, und zwar über deren Auslöseverweise TRIG (siehe Fig. 4). Der Ereignismanager 15 des Klientprogramms KP kann direkt die über die Auslöseverweise TRIG zugeordneten Regelpakete oder Routinen aufrufen, oder über die Listen 7, 8 der lokalen Datenbank DB, in welche die Regelpakete RP ihre Auslöseverweise TRIG eingetragen haben. Auch ist es möglich, nach einem Auslösen des Ereignismanagers 15 bereits im Block 9 des Klientprogramms KP jene Regelpakete - entsprechend ihren Auslöseverweisen TRIG - vorzuselektieren, welche der Ereignisauslösung des Ereignismanagers 15 entsprechen, und anschließend das Klientprogramm 15 nur für diese vorselektierten Regelpakete RP zu durchlaufen.

Eine andere Möglichkeit ist es, die Auslöseverweise TRIG in den Regelpaketen RP als „Filter" für die Abarbeitung der Regelpakete im Zuge einer ereignisgesteuerten Ausführung des Klientprogramms KP zu verwenden: Wenn ein Regelpaket zumindest einen Auslöseverweis TRIG enthält, wird es bei einem Aufruf durch das Klientprogramm KP nur dann ausgeführt, wenn auch sein Auslöseverweis TRIG diesem Ereignis entspricht.

Fig. 10 zeigt ein Implementierungsbeispiel des Klientprogramms KP im Rahmen eines Betriebssystems eines Benutzerrechners 2, welches geschützte Bereiche („Kontexte") für einzelne Prozesse bereitstellt, beispielsweise um Benutzerprozesse von Systemprozessen zu isolieren und dadurch die Betriebsstabilität zu verbessern. Da die Installation und Konfiguration von Softwarekomponenten häufig kontextübergreifende Berechtigungen erfordert, wird hier der Abarbeitungsprozeß P in mehrere in geschützten Systembereichen „Admin", „User" und „System" ablaufende Ausführungsmaschinen P_1, P_2 und P_3 unterteilt.

Für jede systemmodifizierende Komponente des Verfahrens, beispielsweise die Routinen der Regelpakete, kann ein Transak-

tionssystem implementiert werden, welches einen vollständigen Rollback der Systemkonfiguration ermöglicht, falls die Installation, Deinstallation, Konfiguration oder Dekonfiguration einer Softwarekomponente fehlschlägt.

Die Erfindung ist nicht auf die dargestellten Ausführungsformen beschränkt, sondern umfaßt alle Varianten und Modifikationen, die in den Rahmen der angeschlossenen Ansprüche fallen.

ANLAGE 1

```
[sml::header]
smlversion=3.0.0
encoding=iso-8859-1
type=dsf
objectid=3-A11-100A-57F0-1000C000001-0-0
sqn=3-FFFF-100A-57F0-25-0-0
name=Microsoft Office XP Premium Edition

[dsf::query_exist]
/*   detect Office 10.0 components   */
if.reg.keyexist condition:true,-
>reghive=HKEY_LOCAL_MACHINE,regpath=SOFTWARE\Microsoft\Office\10.0,
{
   do.reg.setkey
regkeyhandle:hRegistry,reghive=HKEY_LOCAL_MACHINE,regpath=SOFTWARE\Microsoft\Office\10.0\Wor
d\InstallRoot,option=OPEN,
   if.sys.handleisvalid condition:true regkeyhandle:hRegistry,
   {
      ;detect WinWord
      ;first let's see if the required file exists
      if.file.exist condition:false,->
      .->filepath=<!--get.reg.value regkeyhandle:hRegistry,->regentry=Path,--!>WINWORD.EXE,
      {
         do.sys.exit->level=section,returnvalue=false,
      }
      ;second let's check the files required property
      if.file.matchversionproperty condition:false,->
      .->filepath=<!--get.reg.value regkeyhandle:hRegistry,->regentry=Path,--!>WINWORD.EXE,
      .->propertyname=fileversion, versionproperty type:eq,=10.*,
      {
         do.sys.exit->level=section,returnvalue=false,
      }
      do.sys.closehandle regkeyhandle:hRegistry,
   }
}

[dsf::query_active]
/*   dedect if Office 10.0 WinWord component is running   */
if.reg.keyexist condition:true,-
>reghive=HKEY_LOCAL_MACHINE,regpath=SOFTWARE\Microsoft\Office\10.0,
{
   do.reg.setkey
regkeyhandle:hRegistry,reghive=HKEY_LOCAL_MACHINE,regpath=SOFTWARE\Microsoft\Office\10.0\Wor
d\InstallRoot,option=OPEN,
   if.sys.handleisvalid condition:true regkeyhandle:hRegistry,
   {
      ;dedect WinWord
      if.process.exist condition:false,->processname=WINWORD.EXE, module=<!--get.reg.value
regkeyhandle:hRegistry,->regentry=Path,--!>WINWORD.EXE,
      {
         call.sys.exit->level=section,returnvalue=false,
      }
      call.sys.closehandle regkeyhandle:hRegistry,
   }
}
```

ANLAGE

[sml::header]
smlversion=3.0.0
encoding=iso-8859-1
type=psf
objectid=3-3F1-100A-57F0-1000C000001-0-0
sqn=3-FFFF-100A-57F0-2-0-0
name=Microsoft Office XP Premium

[psf::definition]
packagename->text=Microsoft Office XP,
packagedescription->text=The Microsoft Office XP Premium Suite,
packagecompany->text=Microsoft,
packagecopyright->text=Copyright© Microsoft Corporation 1985-2001. All rights reserved.,
packageproductversion->versionnumber=10.0,
packagedate->text=2002-01-01,

[sml::system]
transactioncontext->context=package,
securitycontext->context=ADM,
oscontext->context=Win32,

[sml::ossupport]
windowssys->platform=x86,os=nt,osversion type:==,=5.0,sp type:>=,=3,
windowssys->platform=x86,os=nt,osversion type:==,=5.1,sp type:>=,=1,
winesys->platform=x86,wineversion type:>=,=2.0.0,winetype=CROSSOVER_OFFICE,

[psf::detectself]
detectsoftware->objectid=3-A11-100A-57F0-1000C000001-0-0,

[sml::displaysupport]
display->show=1,
displayheader->text=Microsoft Office XP,
displaytext->text=Manages Microsoft Office XP...,

[psf::archive]
archivepolicies->archive=1,override=1,
archiveuninstall->archive=1,

[psf::installoptions]
rollbackonerror->rollback=1,
installevents->event=ALL,
ownersonly->restrict=0,

[psf::dedecttargets]
if.group.accountismember->groupname groupformat:default, type:eq,=officexp,

[psf::install]
installjobid->objectid=3-411-100A-57F0-1000C000001-0-0,

[psf::uninstall]
uninstalljobid->objectid=3-441-100A-57F0-1000C000001-0-0,

[psf::policies_true]
policyid->objectid=3-471-100A-57F0-1000C000001-0-0,

[psf::policies_false]
policyid->objectid=3-471-100A-57F0-1000C000002-0-0

Patentansprüche:

1. Verfahren zur automatischen Installation un Konfiguration von Softwarekomponenten (SW) in einem Computer netzwerk (1), das eine Vielzahl von Benutzerrechnern (2) un zumindest eine Netzwerkressource (RES) von installierbare Softwarekomponenten umfaßt, wobei die erfolgreiche Installatio einer Softwarekomponente die An- oder Abwesenheit einer andere Softwarekomponente voraussetzen kann, gekennzeichnet durch di Schritte:

a) Bereitstellen eines Frameworks (FW) auf der Netzwerk ressource (RES), welches ein Regelpaket (RP) für jede instal lierbare Softwarekomponente (SW), einen Detektor (DET) für jed mögliche Voraussetzung und eine Liste (L) abzuarbeitender Re gelpakete (RP) umfaßt,

wobei jedes Regelpaket (RP) zumindest eine der folgende vier Routinen umfaßt: eine Routine (4) zum Installieren de Softwarekomponente (SW) auf dem Benutzerrechner (2), eine Rou tine (4') zum Deinstallieren derselben, eine Routine (5) zu Konfigurieren der installierten Softwarekomponente (SW) un eine Routine (5') zum Rückgängigmachen (Dekonfigurieren) de Konfiguration hinsichtlich derselben;

b) Übertragen des Frameworks (FW) an einen Benutzerrech ner (2);

c) Abarbeiten der Liste (L) abzuarbeitender Regelpaket (RP) auf dem Benutzerrechner (2) unter Aufrufen ihrer Installa tionsroutinen (4), und nochmaliges Abarbeiten der Liste (L) ab zuarbeitender Regelpakete auf dem Benutzerrechner (2) unte Aufrufen ihrer Konfigurationsroutinen (5);

wobei, wenn im Zuge eines Regelpakets (RP) mittels eine Detektors (DET) festgestellt wird, daß die An- oder Abwesenhei einer anderen Softwarekomponente (SW) erforderlich ist, die In stallations- bzw. Deinstallationsroutine (4, 4') des dieser an deren Softwarekomponente (SW) zugeordneten Regelpakets (RP aufgerufen wird.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß das Framework (FW) auch Detektoren (DET_{HW}, DET_{BS1}) für die Hardware- oder Betriebssystemausstattung eines Benutzerrechners (2) umfaßt und im Zuge einer Routine (4, 4', 5, 5') mittels eines solchen Detektors überprüft wird, ob der Benutzerrechner (2) für die jeweilige Installation, Deinstallation, Konfiguration oder Dekonfiguration der Softwarekomponente (SW) geeignet ist.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß im Zuge einer Routine (4, 4', 5, 5') vorab geprüft wird, ob die jeweilige Installation, Deinstallation, Konfiguration oder Dekonfiguration der Softwarekomponente (SW) auf dem Benutzerrechner (2) bereits erfolgt ist und, wenn ja, die Routine sofort beendet wird.

4. Verfahren nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß Schritt b) und/oder Schritt c) durch ein lokales Ereignis (16-19) auf dem jeweiligen Benutzerrechner (2) ausgelöst wird, z.B. ein Systemereignis wie ein Systemstart oder -stop, eine Benutzeran- oder -abmeldung, eine Netzwerkan- oder -abmeldung, ein Programmstart oder -ende usw., ein Hardwareereignis wie das An- oder Abschließen einer Hardwareausstattung, usw.

5. Verfahren nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß Schritt b) und/oder Schritt c) durch ein entferntes Ereignis auf der Netzwerkressource ausgelöst wird, z.B. das Aussenden einer Gruppen- oder Broadcastnachricht usw.

6. Computerprogramm, implementierend ein Verfahren nach einem der Ansprüche 1 bis 5.

7. Regelpaket, das auf einem Betriebssystem eines Benutzerrechners (2) ausführbar ist, zur automatischen Installation und Konfiguration von Softwarekomponenten (SW), die auf einer Netzwerkressource (RES) verfügbar sind, auf dem Benutzerrechner (2), dadurch gekennzeichnet, daß das Regelpaket (RP) einen Verweis (RES_A) auf eine Softwarekomponente auf der Netzwerkressource (RES) und zumindest eine der folgenden vier Routinen um-

faßt: eine Routine (4) zum Installieren dieser Softwarekompc nente (SW) auf dem Benutzerrechner (2), eine Routine (4') zu Deinstallieren dieser Softwarekomponente (SW) vom Benutzerrech ner (2), eine Routine (5) zum Konfigurieren dieser auf dem Be nutzerrechner (2) installierten Softwarekomponente (SW), un eine Routine (5') zum Rückgängigmachen (Dekonfigurieren) de Konfiguration dieser auf dem Benutzerrechner (2) installierte Softwarekomponente (SW), wobei jede Routine (4, 4', 5, 5') wenn sie das Erfordernis der An- oder Abwesenheit einer andere Softwarekomponente (SW) feststellt, zur Installations- bzw Deinstallationsroutine (4, 4') eines dieser anderen Software komponente (SW) zugeordneten anderen Regelpakets (RP) ver zweigt.

8. Regelpaket nach Anspruch 7, dadurch gekennzeichnet daß es einen Verweis (DET_{HW}, DET_{BS1}) auf eine bestimmte Hard ware- und/oder Betriebssystemausstattung eines Benutzerrechner (2) umfaßt und anhand dieses Verweises überprüft, ob der Benut zerrechner (2) für die jeweilige Installation, Deinstallation Konfiguration oder Dekonfiguration der Softwarekomponente (SW geeignet ist.

9. Regelpaket nach Anspruch 7 oder 8, dadurc gekennzeichnet, daß es überprüft, ob die jeweilige Installa tion, Deinstallation, Konfiguration oder Dekonfiguration de Softwarekomponente (SW) auf dem Benutzerrechner (2) bereits er folgt ist und, wenn ja, seine Ausführung beendet.

10. Regelpaket nach einem der Ansprüche 7 bis 9, dadurc gekennzeichnet, daß es mindestens einen Auslöseverweis (TRIG auf ein lokales Ereignis (16-19) auf dem Benutzerrechner (2 oder ein entferntes Ereignis auf der Netzwerkressource enthält wobei der Auslöseverweis (TRIG) diesem Ereignis zumindest ein der Routinen (4, 4', 5, 5') des Regelpakets zuordnet.

11. Regelpaket nach einem der Ansprüche 7 bis 10, dadurc gekennzeichnet, daß es in einen inaktiven Zustand versetzba ist, in welchem nur seine Deinstallations- und Dekonfigurati onsroutinen (4', 5') aufrufbar sind.

12. Computer, der mit zumindest einem Regelpaket nach einem der Ansprüche 7 bis 11 programmiert ist.

13. Framework, das auf einer Netzwerkressource (RES) in einem Computernetzwerk (1) für eine Vielzahl von Benutzerrechnern (2) bereitstellbar ist, zur automatischen Installation und Konfiguration von auf der Netzwerkressource (RES) verfügbaren Softwarekomponenten (SW) auf den Benutzerrechnern (2), wobei die erfolgreiche Installation einer Softwarekomponente (SW) die An- oder Abwesenheit einer anderen Softwarekomponente (SW) voraussetzen kann, dadurch gekennzeichnet, daß das Framework (FW) einen Satz von Regelpaketen (RP) nach einem der Ansprüche 7 bis 10, einen Satz von Detektoren (DET) für jede mögliche Voraussetzung, und eine Liste (L) von auf den Benutzerrechnern (2) abzuarbeitenden Regelpaketen (RP) umfaßt.

14. Framework nach Anspruch 13 in Verbindung mit einem Regelpaket nach Anspruch 8, dadurch gekennzeichnet, daß das Framework (FW) auch Detektoren (DET_{HW}, DET_{BS1}) für die Hardware- oder Betriebssystemausstattung eines Benutzerrechners (2) umfaßt und den Regelpaketen (RP) für die genannte Überprüfung zur Verfügung stellt.

15. Computer, der mit einem Framework nach Anspruch 13 oder 14 programmiert ist.

16. Maschinenlesbarer Datenträger, der mit einem Framework nach Anspruch 13 oder 14 programmiert ist.

17. Klientprogramm, das auf einem Benutzerrechner (2) ausführbar ist, zur automatischen Installation und Konfiguration von Softwarekomponenten (SW), die auf einer Netzwerkressource (RES) verfügbar sind, auf dem Benutzerrechner (2), dadurch gekennzeichnet, daß es ein Framework (FW) nach Anspruch 13 oder 14 empfängt und speichert, in einem ersten Durchgang die Liste (L) abzuarbeitender Regelpakete (RP) unter Aufrufen ihrer Installationsroutinen (4) und in einem zweiten Durchgang die Liste (L) abzuarbeitender Regelpakete (RP) unter Aufrufen ihrer Konfigurationsroutinen (5) abarbeitet.

18. Klientprogramm nach Anspruch 17, dadurch gekennzeich net, daß es eine lokale Datenbank (DB) aufweist, welche ein Liste (7) von Regelpaketen (RP) mit erfolgreich durchlaufene Installationsroutinen (4) und eine Liste (8) von Regelpakete (RP) mit erfolgreich durchlaufenen Konfigurationsroutinen (5 enthält.

19. Klientprogramm nach Anspruch 18, dadurch gekennzeich net, daß es die in den Listen (7, 8) eingetragenen Regelpaket (RP) mit den im Framework (FW) enthaltenen Regelpaketen (RP vergleicht und für jene Regelpakete (RP), welche im Framewor (FW) nicht aufscheinen, in einem ersten Durchgang deren Dekon figurationsroutinen (5') und in einem zweiten Durchgang dere Deinstallationsroutinen (4') abarbeitet.

20. Klientprogramm nach einem der Ansprüche 17 bis 19 i Verbindung mit einem Regelpaket nach Anspruch 10, dadurch ge kennzeichnet, daß es das Auftreten eines lokalen Ereignisse (16-19) auf dem Benutzerrechner (2), z.B. eines Systemereignis ses wie eines Systemstarts oder -stops, einer Benutzeran- ode -abmeldung, einer Netzwerkan- oder -abmeldung, eines Programm starts oder -endes usw., eines Hardwareereignisses wie des An oder Abschließens einer Hardwareausstattung, usw., und/oder ei nes entfernten Ereignisses auf der Netzwerkressource, z.B. de Aussendens einer Gruppen- oder Broadcastnachricht usw., über wacht und die diesem Ereignis über den Auslöseverweis (TRIG zugeordnete Routine (4, 4', 5, 5') des entsprechenden Regelpa kets (RP) aufruft.

21. Klientprogramm nach einem der Ansprüche 17 bis 20 dadurch gekennzeichnet, daß es ein Transaktionssystem für jed systemmodifizierende Komponente, insbesondere für die Regelpa kete (RP), aufweist.

22. Computer, der mit einem Klientprogramm nach eine der Ansprüche 17 bis 21 programmiert ist.

Zusammenfassung:

Verfahren zur Installation und Konfiguration von Softwarekomponenten

Die Erfindung betrifft ein Verfahren, Regelpaket (RP), Framework (FW) und Klientprogramm (KP) zur automatischen Installation und Konfiguration von Softwarekomponenten (SW) in einem Computernetzwerk (1), das eine Vielzahl von Benutzerrechnern (2) und zumindest eine Netzwerkressource (RES) von installierbaren Softwarekomponenten (SW) umfaßt, wobei die erfolgreiche Installation einer Softwarekomponente (SW) die An- oder Abwesenheit einer anderen Softwarekomponente (SW) voraussetzen kann, sowie entsprechend programmierte Computer und Datenträger. Insbesondere weist jedes Regelpaket (RP) zumindest eine der folgenden vier Routinen auf: eine Routine (4) zum Installieren der Softwarekomponente (SW) auf dem Benutzerrechner (2), eine Routine (4') zum Deinstallieren derselben, eine Routine (5) zum Konfigurieren der installierten Softwarekomponente (SW) und eine Routine (5') zum Rückgängigmachen (Dekonfigurieren) der Konfiguration derselben, wobei die Regelpakete (RP) auf dem Benutzerrechner (2) unter Aufrufen ihrer Installationsroutinen (4) und anschließend ihrer Konfigurationsroutinen (5) abgearbeitet werden.

(Fig. 2)

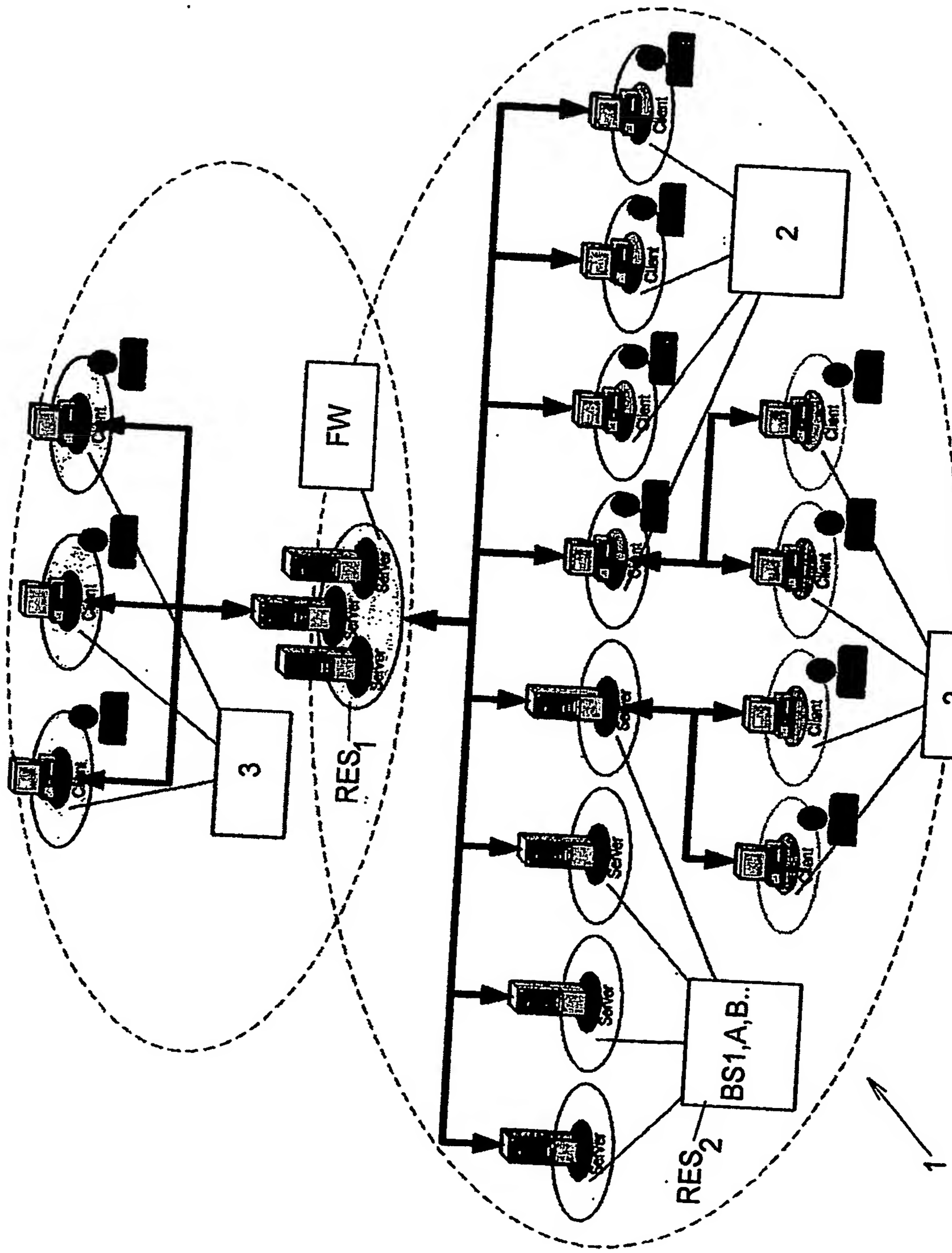


Fig. 1

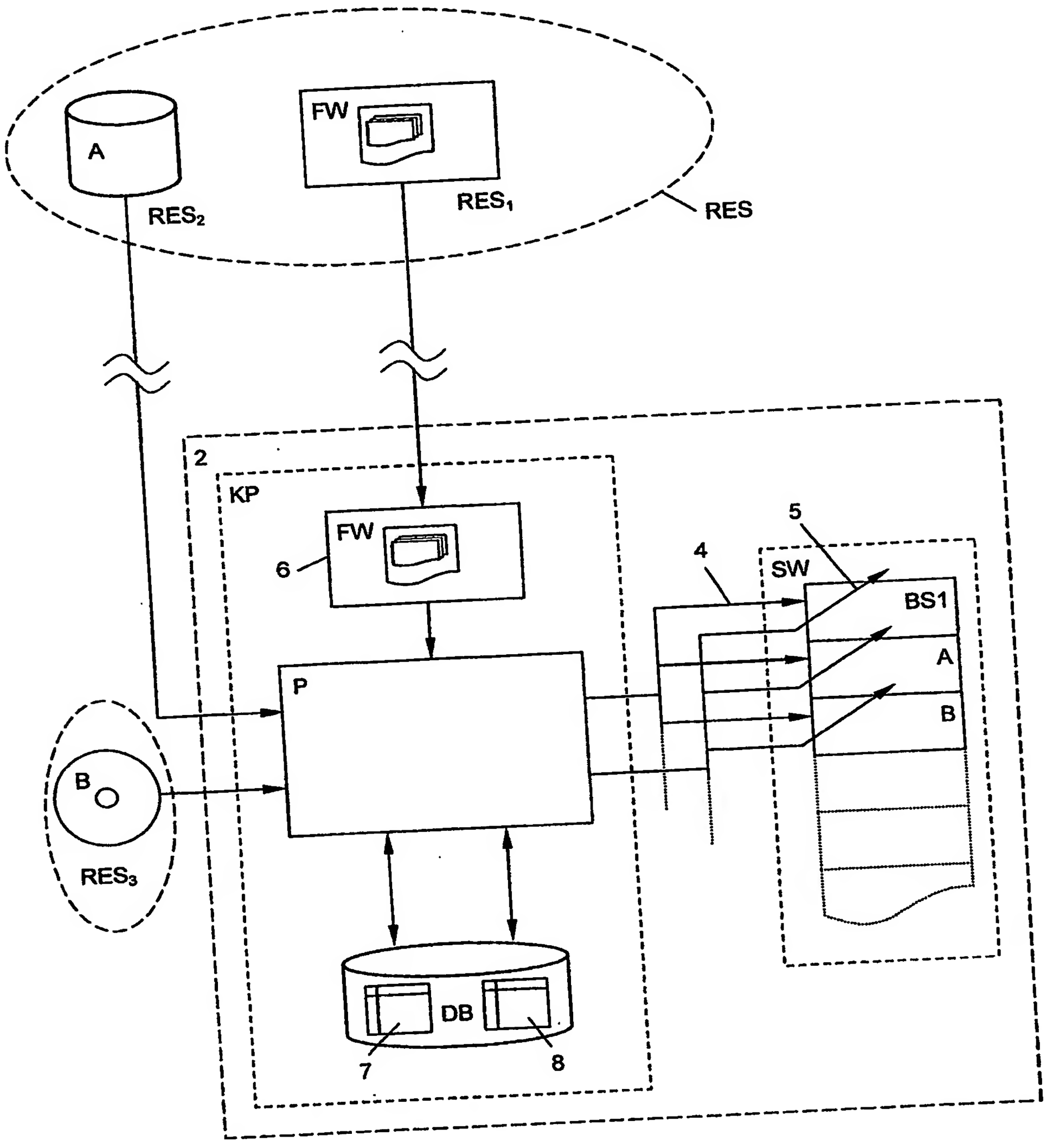


Fig. 2

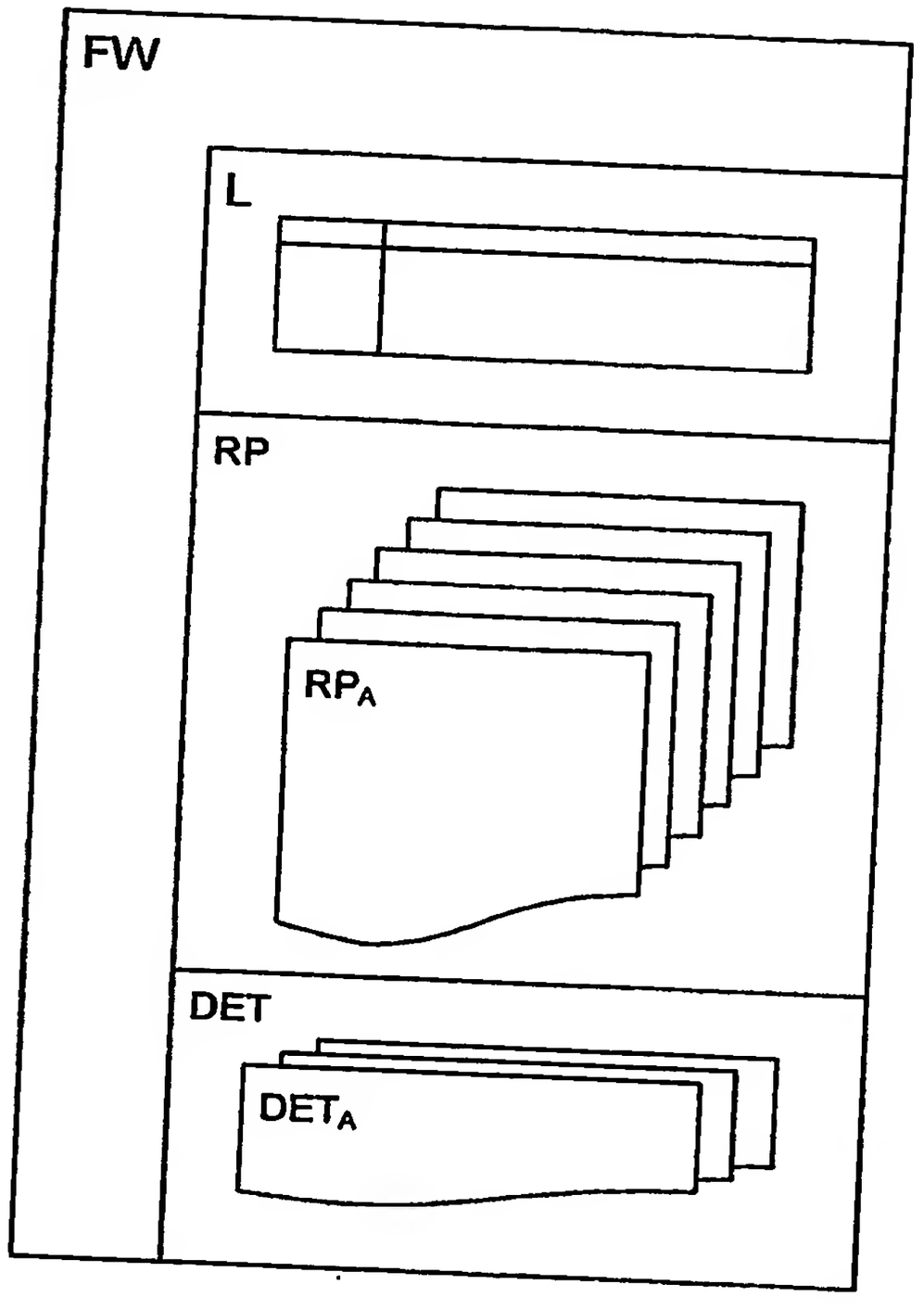


Fig. 3

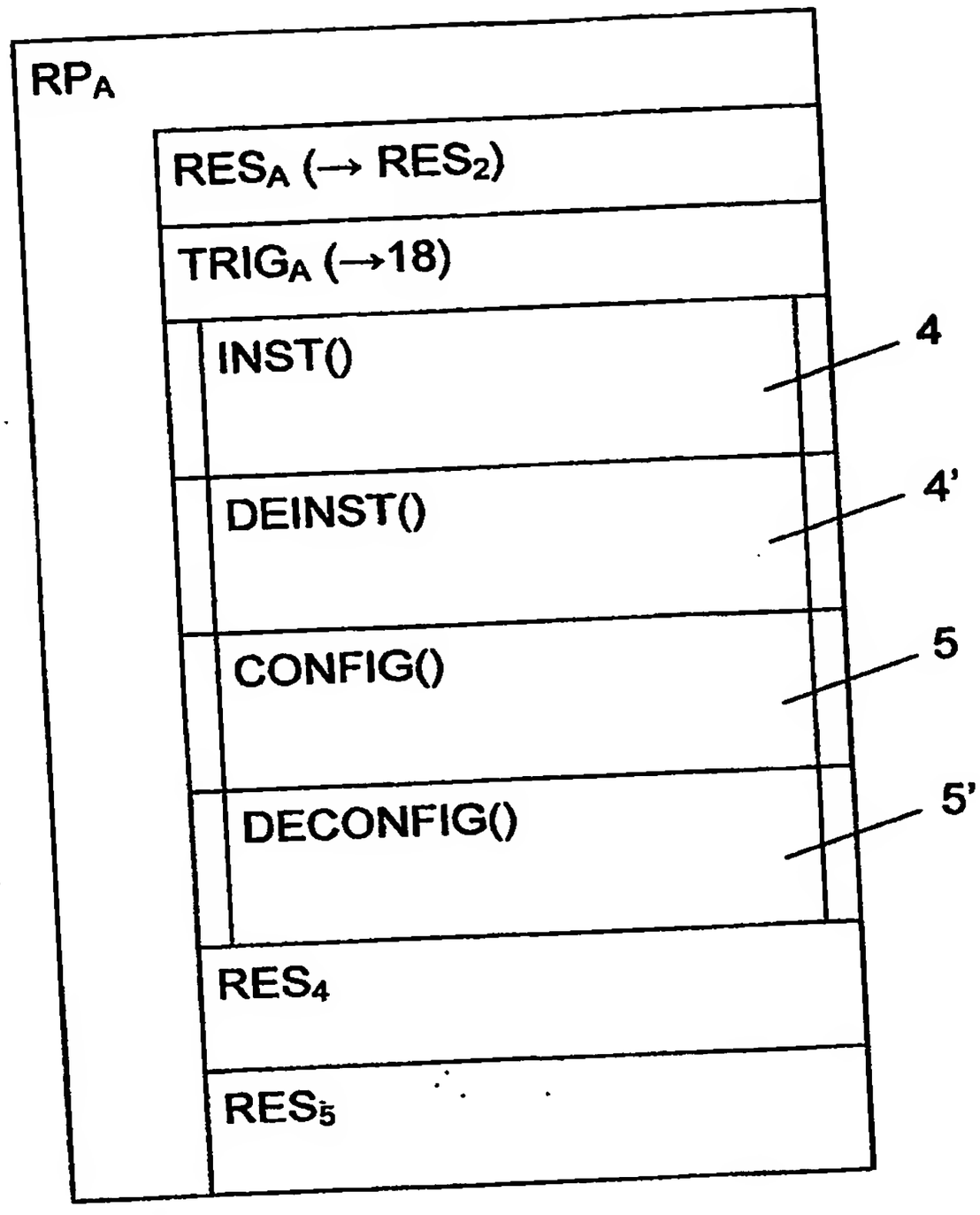

RP$_A$
RES$_A$ (→ RES$_2$)
TRIG$_A$ (→18)
INST()
4
DEINST()
4'
CONFIG()
5
DECONFIG()
5'
RES$_4$
RES$_5$

Fig. 4

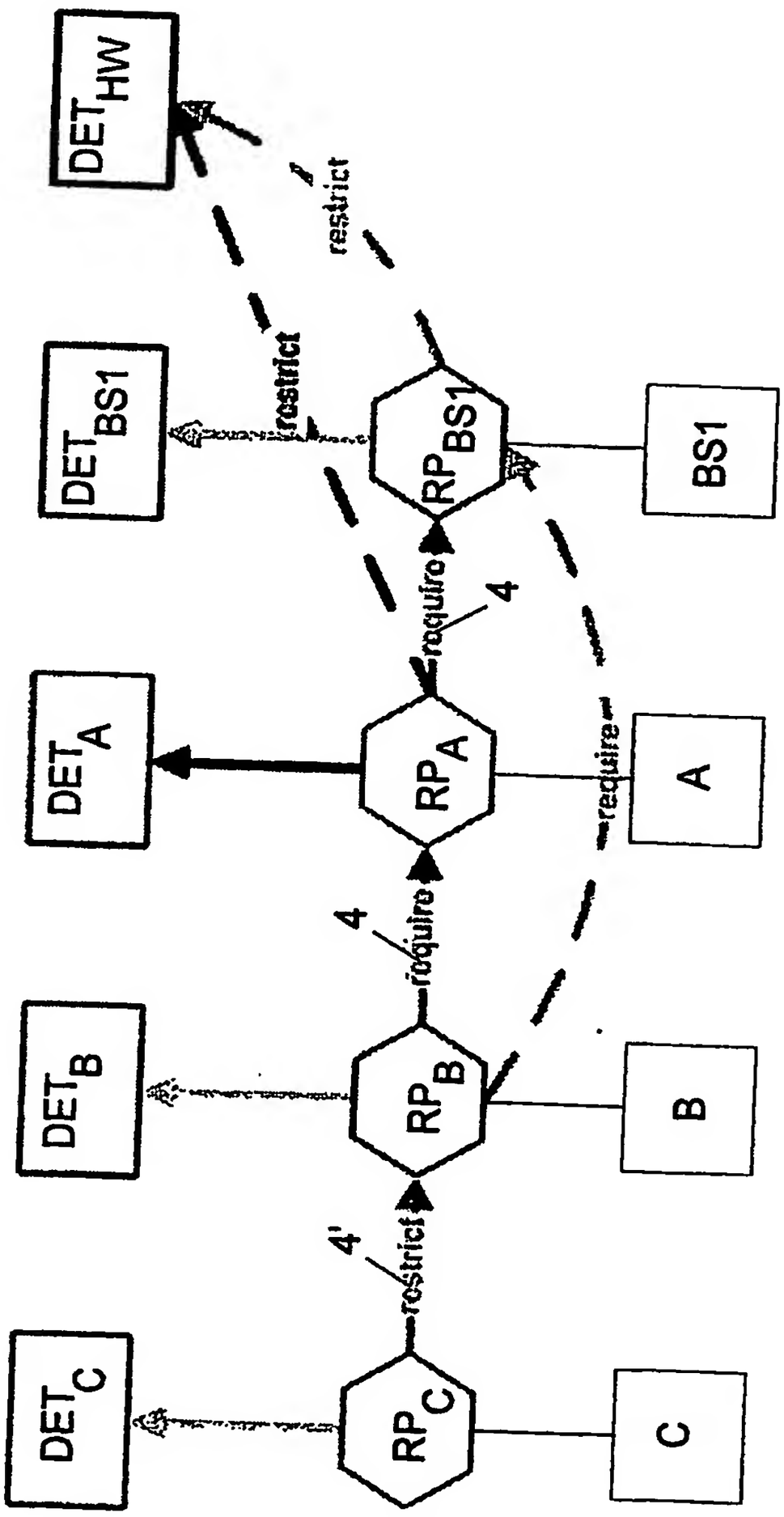

DET HW
restrict
DET BS1
restrict
RP BS1
BS1
require
4
DET A
RP A
A
require
require
4
DET B
RP B
B
4'
restrict
DET C
RP C
C

Fig. 5

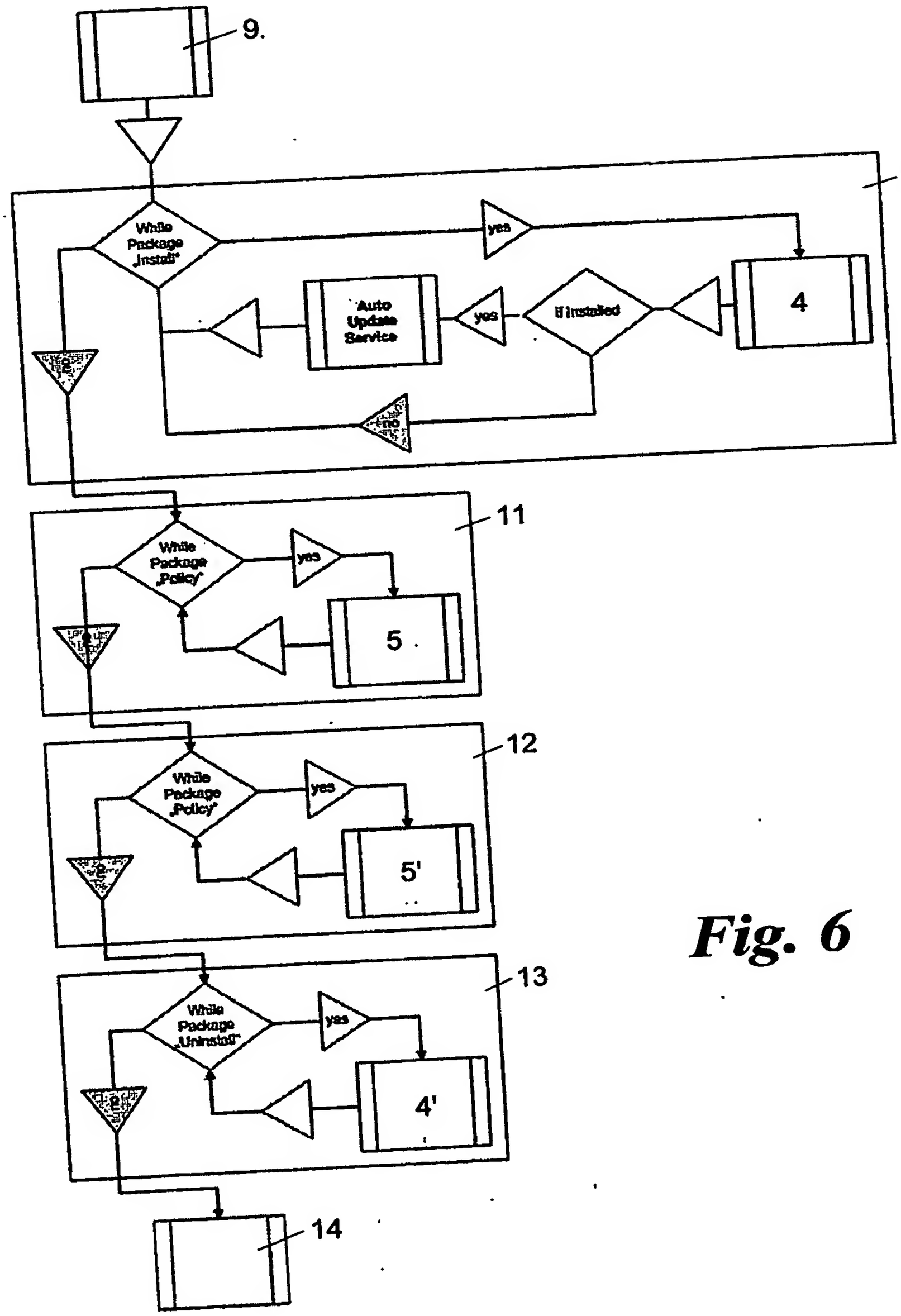

9.
10
While Package „Install"
yes
Auto Update Service
yes
If installed
4
no
11
While Package „Policy"
yes
5
12
While Package „Policy"
yes
5'
13
While Package „Uninstall"
yes
4'
14

Fig. 6

Fig. 7

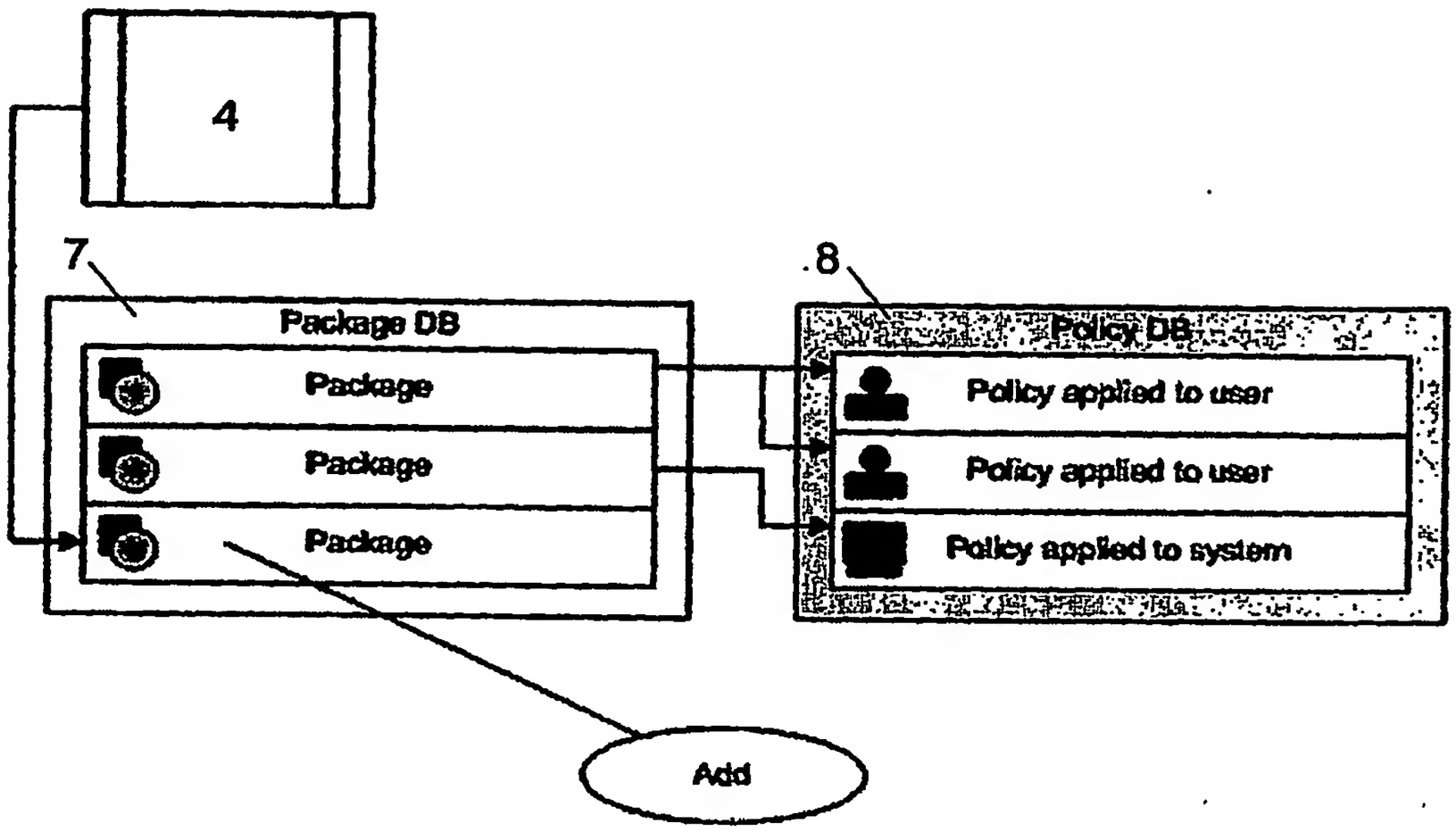
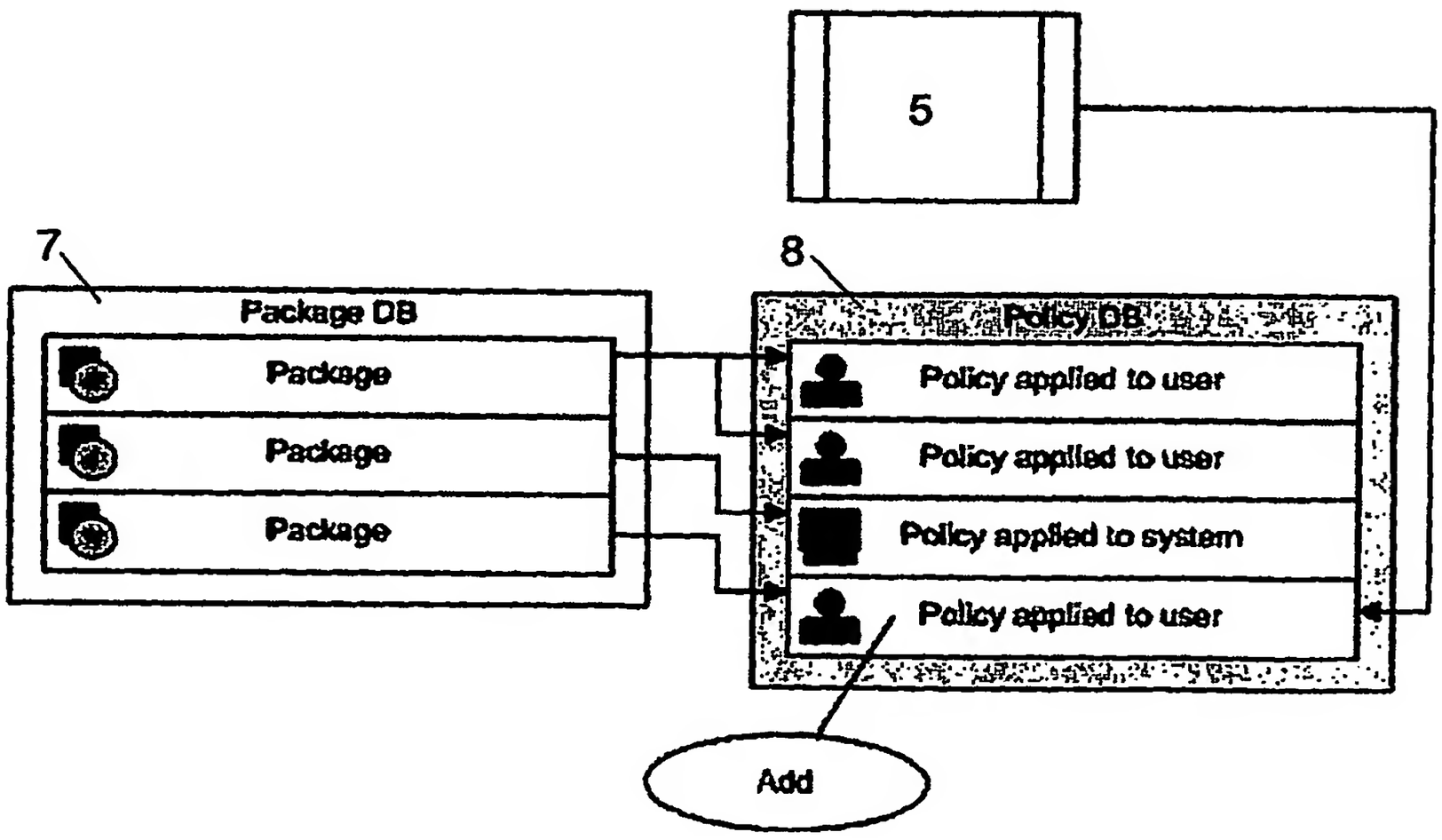

4
7
8
Package DB
Package
Package
Package
Policy DB
Policy applied to user
Policy applied to user
Policy applied to system
Add

Fig. 8


5
7
8
Package DB
Package
Package
Package
Policy DB
Policy applied to user
Policy applied to user
Policy applied to system
Policy applied to user
Add

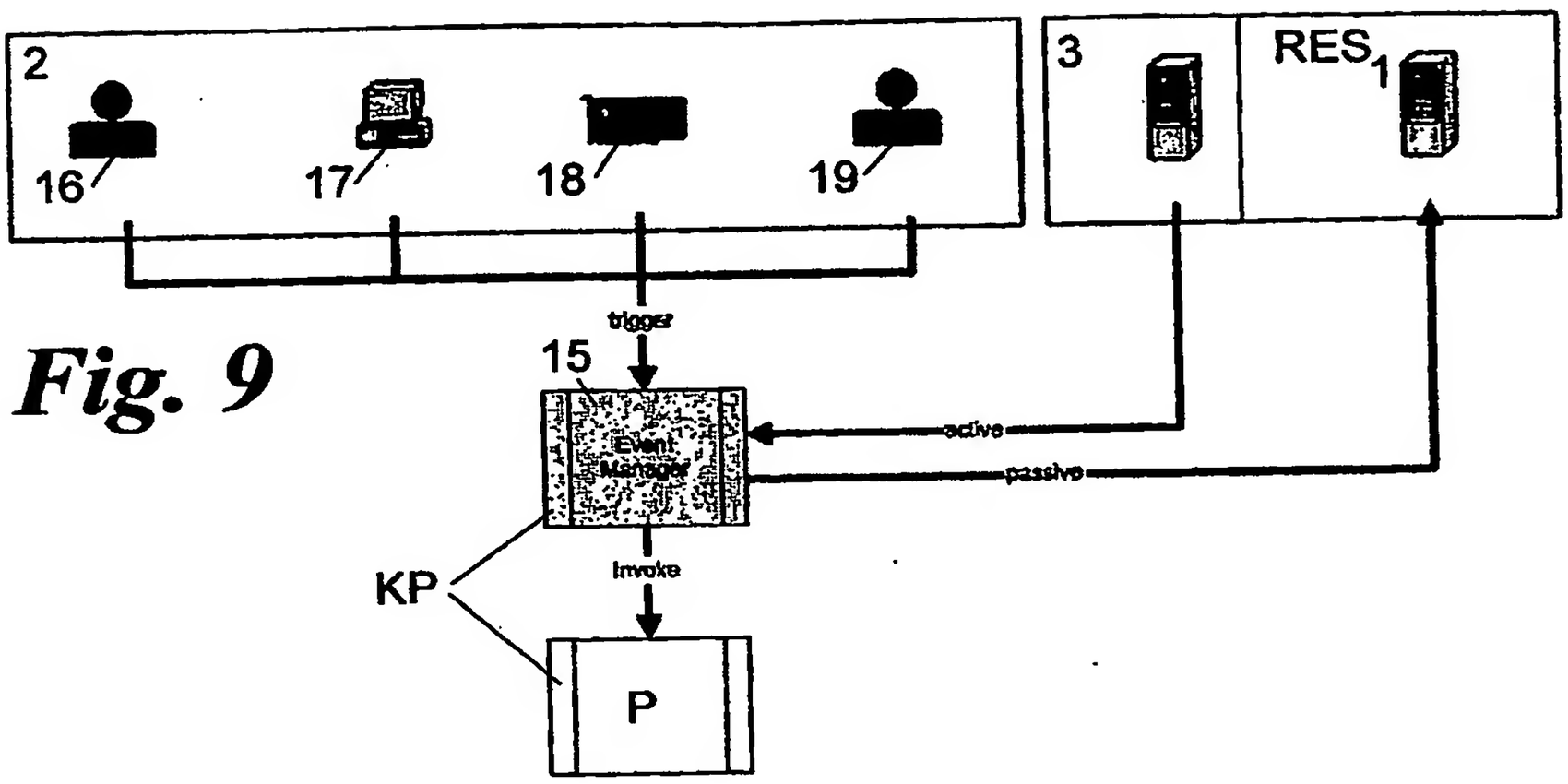
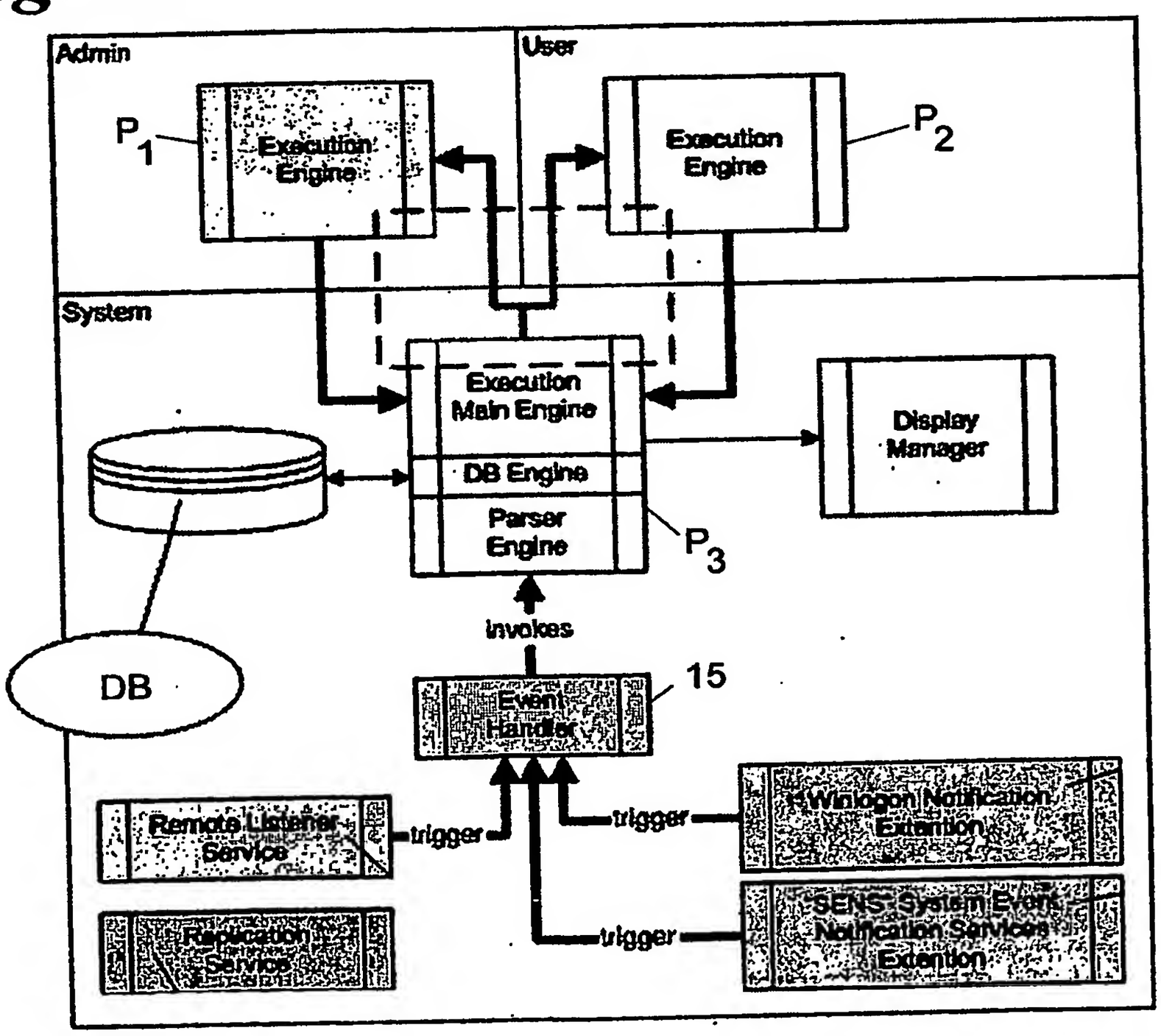

2
16
17
18
19
3
RES1
trigger
15
Event Manager
active
passive
KP
Invoke
P

Fig. 9

Fig. 10


Admin
User
P1
Execution Engine
Execution Engine
P2
System
Execution Main Engine
DB Engine
Parser Engine
P3
Display Manager
DB
invokes
Event Handler
15
Remote Listener Service
trigger
trigger
trigger
Winlogon Notification Extention
SENS System Event Notification Services Extention
Replication Service